

CLAIMS

What is claimed is:

- 1 1. A method comprising:
 - 2 compiling a function including a byte code sequence having a field byte
 - 3 code that accesses or modifies a field, the compiled function providing a native
 - 4 code and occupying a code space;
 - 5 generating an instrumentation code corresponding to a field watch of the
 - 6 field;
 - 7 guarding execution of the instrumentation code if the field watch is not
 - 8 activated; and
 - 9 inserting the instrumentation code to the native code.
- 1 2. The method of claim 1 wherein generating the instrumentation
- 2 code comprises:
 - 3 executing a field watch sequence.
- 1 3. The method of claim 2 wherein guarding execution of the
- 2 instrumentation code comprises:
 - 3 comparing a flag with a predetermined watch value to determine if the
 - 4 field watch is activated.

5 executing an event hook function for an event corresponding to the field
6 watch; and
7 restoring the live global state.

1 9. The method of claim 8 wherein saving the live global state
2 comprises:
3 pushing the live global state onto a stack.

1 10. The method of claim 8 wherein executing the event hook function
2 comprises:
3 passing an argument corresponding to the field; and
4 calling a run-time library function related to the event.

1 11. The method of claim 9 wherein restoring the live global state
2 comprises:
3 retrieve the live global state from the stack.

1 12. The method of claim 3 further comprising:
2 activating the field watch by setting the flag; and
3 clearing the field watch by resetting the flag.

1 13. The method of claim 1 wherein the function is a Java method.

09323105-033001
T00020-507E2860

1 14. The method of claim 1 wherein the field is a Java field in a Java
2 virtual machine.

1 15. The method of claim 8 wherein the event hook function is
2 compatible with a Java Virtual Machine Debug Interface (JVMDI).

1 16. A computer program product comprising:
2 a machine useable medium having computer program code embedded
3 therein, the computer program product having:
4 computer readable program code to compile a function including a
5 byte code sequence having a field byte code that accesses or modifies a
6 field, the compiled function providing a native code occupying a code
7 space;
8 computer readable program code to generate an instrumentation
9 code corresponding to a field watch of the field;
10 computer readable program code to guard execution of the
11 instrumentation code if the field watch is not activated; and
12 computer readable program code to insert the instrumentation code
13 to the native code.

1 17. The computer program product of claim 16 wherein the computer
2 readable program code to generate the instrumentation code comprises:
3 computer readable program code to execute a field watch sequence.

1 18. The computer program product of claim 17 wherein the computer
2 readable program code to guard execution of the instrumentation code comprises:

3 computer readable program code to compare a flag with a predetermined
4 watch value to determine if the field watch is activated.

1 19. The computer program product of claim 18 wherein the computer
2 readable program code to insert the instrumentation code comprises:

3 computer readable program code to insert the instrumentation code before
4 a field access or modification point.

1 20. The computer program product of claim 17 wherein the computer
2 readable program code to insert the instrumentation code comprises:

3 computer readable program code to insert the instrumentation code at end
4 of the code space.

1 21. The computer program product of claim 20 wherein the computer
2 readable program code to guard execution of the instrumentation code comprises:

3 computer readable program code to update an offset of a jump instruction
4 to a stub having the field watch sequence when the field watch is activated.

1 22. The computer program product of claim 20 wherein the computer
2 readable program code to guard execution of the instrumentation code comprises:

3 computer readable program code to replace a no-op sequence with a jump
4 instruction to a stub having the field watch sequence when the field watch is
5 activated.

1 23. The computer program product of claim 17 wherein the computer
2 readable program code to execute the field watch sequence comprises:

3 computer readable program code to save live global state, the live global
4 state corresponding to an active register;

5 computer readable program code to execute an event hook function for an
6 event corresponding to the field watch; and

7 computer readable program code to restore the live global state.

1 24. The computer program product of claim 23 wherein the computer
2 readable program code to save the live global state comprises:

3 computer readable program code to push the live global state onto a stack.

1 25. The computer program product of claim 23 wherein the computer
2 readable program code to execute the event hook function comprises:

3 computer readable program code to pass an argument corresponding to the
4 field; and

5 computer readable program code to call a run-time library function related
6 to the event.

1 26. The computer program product of claim 24 wherein the computer
2 readable program code to restore the live global state comprises:

3 computer readable program code to retrieve the live global state from the
4 stack.

1 27. The computer program product of claim 18 further comprising:
2 computer readable program code to activate the field watch by setting the
3 flag; and
4 computer readable program code to clear the field watch by resetting the
5 flag.

1 28. The computer program product of claim 16 wherein the function is
2 a Java method.

1 29. The computer program product of claim 16 wherein the field is a
2 Java field in a Java virtual machine.

1 30. The computer program product of claim 23 wherein the event hook
2 function is compatible with a Java Virtual Machine Debug Interface (JVMDI).

1 31. A system comprising:
2 a processor;
3 a memory coupled to the processor, the memory storing instruction code,
4 the instruction code, when executed by the processor, causing the processor to:

5 compile a function including a byte code sequence having a field
6 byte code that accesses or modifies a field, the compiled function
7 providing a native code and occupying a code space,
8 generate an instrumentation code corresponding to a field watch of
9 a field,
10 guard execution of the instrumentation code if the field watch is
11 not activated, and
12 insert the instrumentation code to the native code.

1 32. The system of claim 31 wherein the instruction code causing the
2 processor to generate the instrumentation code causes the processor to:

3 execute a field watch sequence.

1 33. The system of claim 32 wherein the instruction code causing the
2 processor to guard execution of the instrumentation code causes the processor to:

3 compare a flag with a predetermined watch value to determine if the field
4 watch is activated.

1 34. The system of claim 33 wherein the instruction code causing the
2 processor to insert the instrumentation code causes the processor to:

3 insert the instrumentation code before a field access or modification point.

1 35. The system of claim 32 wherein the instruction code causing the
2 processor to insert the instrumentation code causes the processor to:

3 insert the instrumentation code at end of the code space.

1 36. The system of claim 35 wherein the instruction code causing the
2 processor to guard execution of the instrumentation code causes the processor to:

3 update an offset of a jump instruction to a stub having the field watch
4 sequence when the field watch is activated.

1 37. The system of claim 35 wherein the instruction code causing the
2 processor to guard execution of the instrumentation code causes the processor to:

3 replace a no-op sequence with a jump instruction to a stub having the field
4 watch sequence when the field watch is activated.

1 38. The system of claim 32 wherein the instruction code causing the
2 processor to execute the field watch sequence causes the processor to:

3 save live global state, the live global state corresponding to an active
4 register;

5 execute an event hook function for an event corresponding to the field
6 watch; and

7 restore the live global state.